One of the outcomes of the Internet worm incident of 1988 was the funding by the federal government of a national Computer Emergency Response Team (CERT) as a clearing house for information about computer vulnerabilities and incidents. By 1995 when formal statistics on vulnerability reporting began, CERT was seeing 171 vulnerabilities. This has grown by leaps and bounds, with the 2005 number being 5990. The computer systems continue to have very large numbers of vulnerabilities.

It was in response to this situation in the late 1970s and 1980s that the need for computer intrusion detection became clear. It was discovered in practice that it was extremely difficult to build perfect security into marketable computer systems, and it was already obvious that intruders were going to cause serious harm by exploiting the resulting vulnerabilities.

IId) Background on Intrusion Detection

The first paper to discuss the possibility of automated computer intrusion detection was a report by James Anderson in 1980 [And80] for the US government. Anderson considered the problem of manual audit trail inspection which was then being performed for audit trails of an IBM mainframe used for classified computing projects to ensure the security of the computer. This was extremely time consuming. Anderson's paper discusses at length the possible risks to the computer, the deficiencies of the audit trail for detecting attacks on the computer, together with various possible improvements.

Anderson also considered the possibility of automated statistical detection of computer intruders masquerading as other users. He considered such measures as what time of day a given user was accessing the computer, how much resource usage (cpu etc) a given program was using, and which files a user or program was accessing. He postulated finding the average and standard deviations for these various measures, and looking for situations where a user or program was too many standard deviations from the mean of such program executions. He also considered comparing measures against fixed thresholds. Anderson's report contemplated a batch system that would run nightly to process that days' security audit trail.

The Anderson report was an analysis of requirements and some initial design ideas for an audit monitoring system. Such a system had not been built at the time.

The next significant development in intrusion detection research was made by SRI, and SRI's early contributions to the field are worthy of their own section.

IIe) Early Intrusion Detection Research at SRI

Here we look at what SRI developed between the early 1980s when they began work in this area, and the end of the NIDES project in 1995, the last day before the statutory bar for the first of the patents that they filed. All work described here is published prior art for the purposes of SRI's patents. For the sake of clarity, we consider the development of SRI's systems in isolation, but readers should be aware that significant developments

were made by other research groups also, and some of those will be described later in this document.

SRI's research began in the early 1980s with an investigation for the US government of audit records on an IBM mainframe similar to the one considered by Anderson [SRI06]. However, it was not until 1986 that Dorothy Denning published a paper on the IDES (Intrusion Detection Expert System) model. That paper has gone on to be the most widely cited paper ever in the field of computer intrusion detection, with over 800 citations to date. The paper describes an outline design for an intrusion detection system which would operate to detect computer intrusions by inspecting computer audit trails in real-time.

The paper discusses an abstract model for what security audit trails should look like (together with practical deficiencies of various audit trails of the time such as the SMF facility on IBM mainframes, and the recording facilities of Berkeley Unix). An idealized audit trail has a *Subject* (the user or program performing an action), an *Object* (the file, program, etc that was being acted on), the *Action* that was being taken by the subject on the object (essentially equivalent to the system call that was being invoked by the subject program), an *Exception-Condition* field which recorded any errors that might have occurred in carrying out the action, a *Resource-Usage* record of how much computer resources were required to carry out the action, and a *Time-stamp* which records when exactly the action occurred.

Given a stream of such audit records, the IDES paper contemplates various kinds of statistical metrics (alternatively called measures in the paper), including *event counters* which count the instances of some type of audit record, such as "the number of logins during an hour, number of times some command is executed during a login session, and number of password failures during a minute". Then there were *interval timers* which represent the length of time between two related events, and *resource measures* which consider the amount of resources used in some action during a period. Examples "are the total number of pages printed by a user per day and total amount of CPU time consumed by some program during a single execution"

Given a measure constructed out of audit records, the paper then contemplates a variety of *statistical models* for this model. For example, the paper refers to the possibility of comparing the metric to fixed thresholds (as foreshadowed in Anderson's paper), comparing an observation to the mean of past observations of that metric to see whether it was too many standard deviations away (also mentioned in Anderson). However, the IDES implementation seems [Jav91] to have ended up by focusing on using a multivariate approach in which the a short term exponentially aged average of the entire vector of intrusion detection measures was compared to a longer term covariance matrix of past behavior of those measure.

In IDES, a statistical profile consists of a definition of the kind of audit record to be considered, the metric to be used, and the type of model to be applied to determine if a new observation was anomalous relative to the past behavior of the system. A variety of possible profiles based on system audit trails are laid out.

Additionally, IDES had a rule-based component allowing it to look for hard-and-fast scenarios which reliably indicated intrusions.

In the early 1990s, IDES was re-engineered by SRI and renamed NIDES, for Next-Generation Intrusion Detection Expert System. To a significant degree, the changes from NIDES to IDES were engineering and practicality improvements rather than fundamentally new ideas. For example the system now acquired a graphical user interface. The statistical algorithms were changed slightly to make them more robust to realistic distributions of activity.

However, one development of significance to this case is that NIDES was designed to run on a network of computers that were interconnected to one another. The system still worked by examining the audit trails of individual computers, but it now had a client-server architecture, allowing multiple audit trails from multiple computers to be consolidated together and processed. Some of the rules in the rule base began to reason about network activity.

For example ([And94 p89-92]), NIDES in 1994 had rules to detect a variety of network services being used on the machine being audited. Thus if a user made a connection to the login, or remote shell, or mount service, this fact would be noted by NIDES. In some cases, NIDES distinguished between successful connections and unsuccessful connections (eg the RemoteMount1 rule reported successful remote mounting of a file system, while the RemoteMount2 reported cases where the remote mounting failed due to some kind of error).

Further development of NIDES in this direction was contemplated. Eg in [And95], consideration of NIDES being extended to directly read network packets was discussed as a future possibility. The packets would be converted to NIDES audit records in their canonical format. Consideration was given to what kind of statistical measures might be applied to the network records:

> "For the NIDES statistical component, a set of statistical intrusion-detection measures that are specific to network-level intrusion are easily developed. For example, such measures might profile the amount of traffic originating or being received by local and remote hosts for given times of the day and days of the week. This information could be correlated with information received from audit trails, such as port number and network activity type, to build additional measures of interest."

In fact, there was to be a funding changover for SRI's intrusion detection efforts from the Navy to DARPA. There were also some personnel changes at SRI – leaders of the IDES/NIDES development such as Teresa Lunt and Debra Anderson were to leave, and Phillip Porras joined the group. When funding for SRI's intrusion research resumed, under Mr Porras's leadership, the group began another cycle of re-engineering the system, which was now called Emerald (Event Monitoring Enabling Responses to Anomalous Live Disturbances), which we shall discuss later.

For convenience, here is a timeline of the major early research systems developed by SRI and UC Davis.

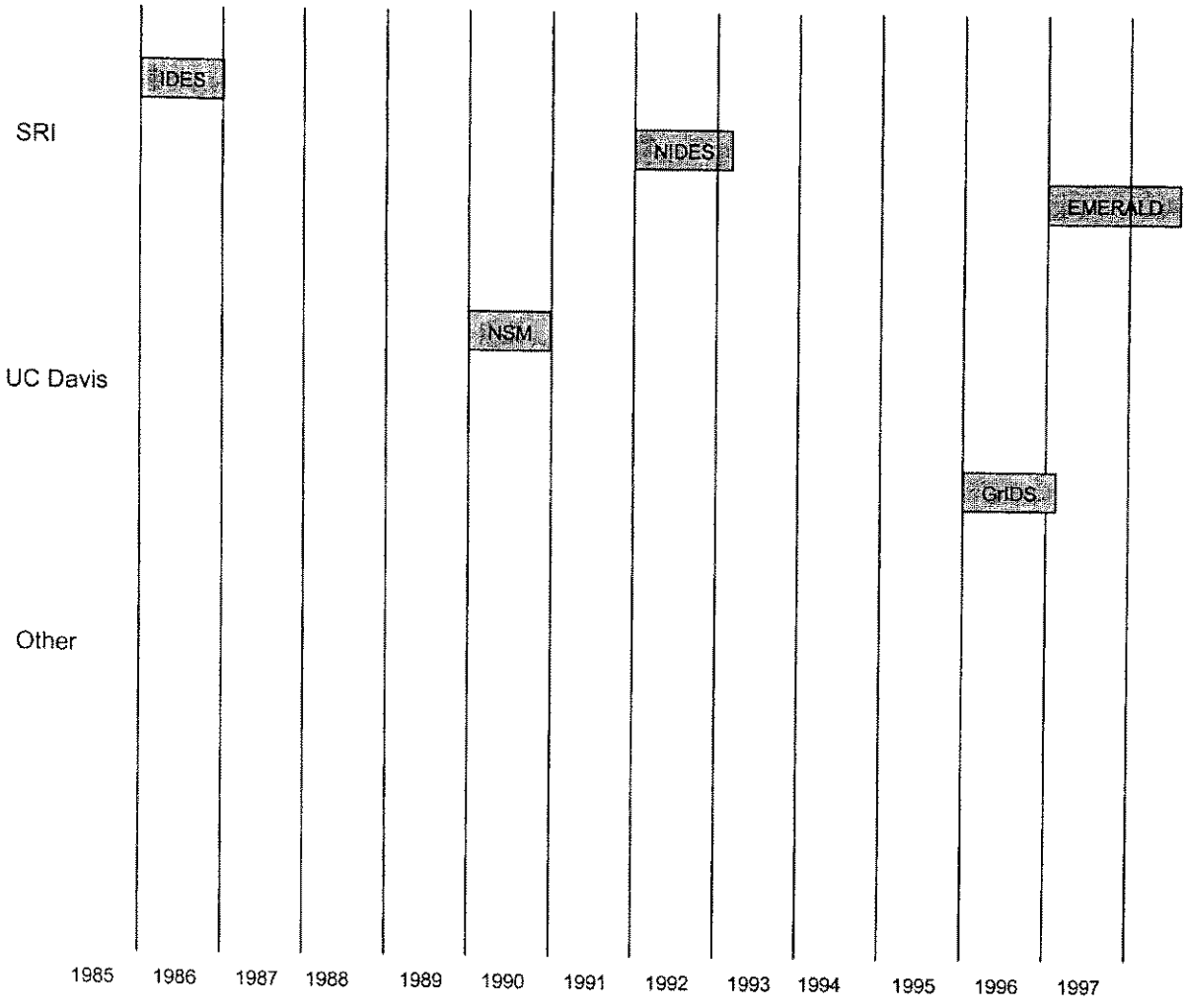

Figure 18: Timeline of major intrusion detection systems discussed in this report

IIf) The NIDES Statistical Algorithm

Because it will be important later, we give here a detailed description of the NIDES algorithm. This builds on our introduction to statistical anomaly detection, and will be important to understanding the issues later. NIDES was more-or-less the state-of-the-art in statistical anomaly detection for detecting intrusions prior to the statutory bar for these patents.

The NIDES statistical algorithm is quite complex and we shall work up to it in several stages. We will start by considering the case of a single measure. For our measure in explaining NIDES we will choose a hypothetical example. Imagine that every time the user on a modern Windows system clicks to change from one application to another, or starts the computer and selects an application, we create an audit record of what the newly selected application was, along with the time of selection and the user doing the selection. Clearly, this data would tell us something about what applications a user makes use of, which in turn would be dependent on their work and leisure activities on their computer.

One user, let us call her Alice, might be a retired homemaker who mainly uses her computer for email with her family, and to check things on the web. As such, almost all of her computer use is in just two applications: Outlook and Internet Explorer (IE). A second user, let us call her Brenda, is a financial analyst and uses a mixture of office applications (Excel, Word, Powerpoint), along with Outlook and IE. She also uses her computer to download music with iTunes.

This kind of measure is called a *categorical measure*, since the variable is drawn from a list of categories (in this case computer applications), rather than being a number.

If we took this measure of the two users averaged over a long period, we might find histograms something like the one here in Figure 19.

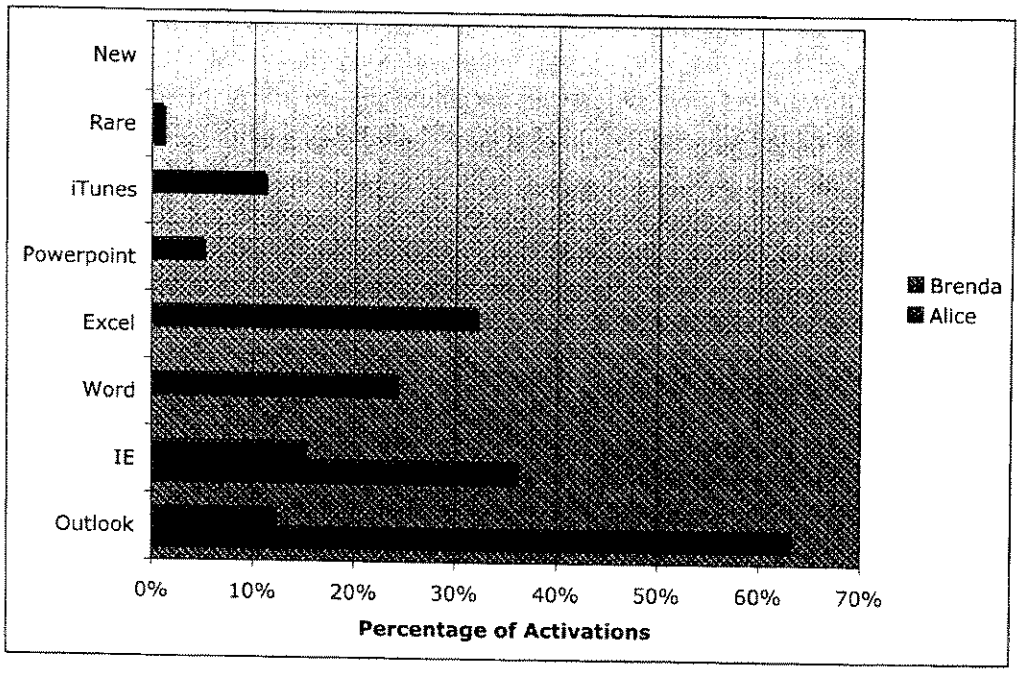

Figure 19: Long term averages for application choices by two hypothetical users

This is a way of representing the *long-term-profile* for these users, or at least that part of the profile associated with this particular measure (Application Activations) – the full profile will involve multiple measures. Clearly, Brenda's profile looks quite different than Alice's. Brenda uses applications that Alice never uses. Alice very occasionally uses some other applications that get lumped together in NIDES into a *rare category*, but neither user ever uses a new application.

On the other hand, if we checked over a shorter period, perhaps about a day, we would find somewhat different frequencies of application usage. Perhaps in this particular period, Brenda was preparing for an important presentation and using PowerPoint much more heavily than on average.

The next graph in Figure 20 adds this particular day of Brenda's activity (her *short-term profile)*, along with her long-term and Alice's.

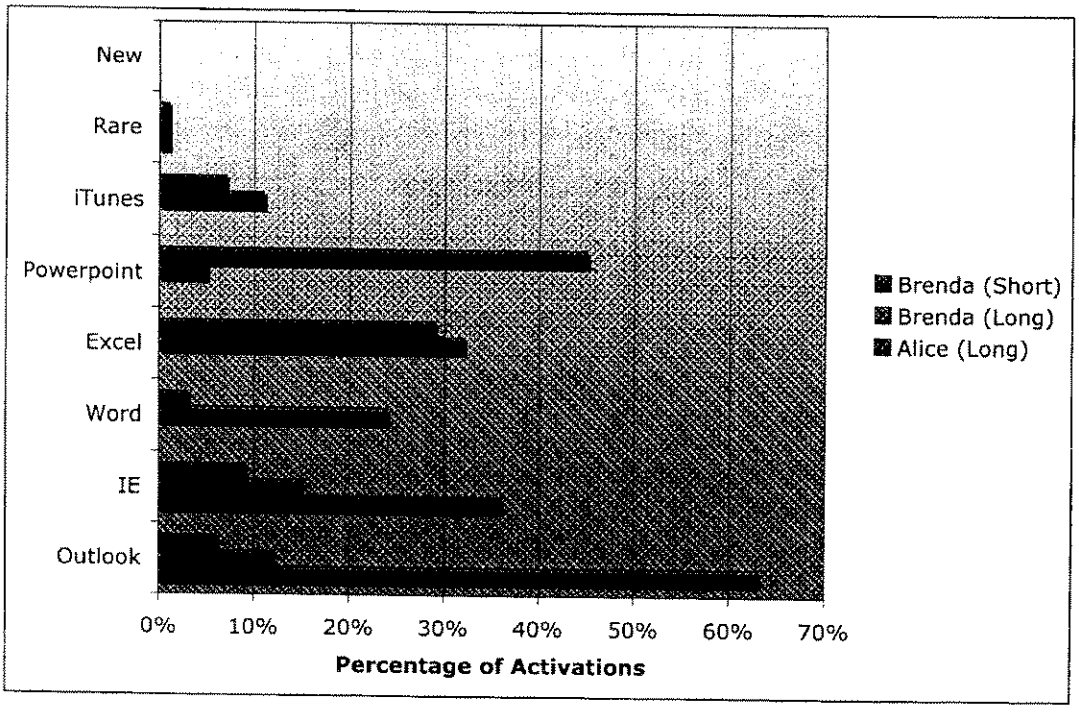

Figure 20: Long term application choice profiles for Alice and Brenda, with short term profile for Alice over some hypothetical period.

Clearly, although Brenda's short-term profile (the blue bars) is different than her long term profile (the plum bars), it still looks more like Brenda than Alice (the green bars). The basic idea of the NIDES statistical algorithm is to develop a way to compare these short-term profiles to the long-term profiles. The hope is to be able to distinguish users from one another, such that, if Jack, an industrial spy, slips into Brenda's office at lunch time and starts using an archive utility to burn DVDs of Brenda's files, while downloading, compiling, and running a backdoor program that will monitor Brenda's keystrokes, these unusual program invocations will change Brenda's short term profile enough that an intrusion detection system can detect that someone is *masquerading* as Brenda.[12]

Let us now explore in more detail how NIDES built statistical profiles and compared them, starting with the long term profile.

The long-term profile is updated every night, and involves incorporating the distribution of categories that occurred during that day. The way this is done is via a technique called *exponential averaging* that allows efficient incorporation of new information into a running average without keeping the entire history around to recompute the average.

[12] Note that the NIDES algorithm has not achieved widespread commercial success for, but it was quite influential in the research community.

Exponential averaging also causes profiles to adapt smoothly and gradually to changes in user behavior.

The idea is that we take the old profile, multiply all the columns down by a value that is a bit less than one, take the days histogram, multiply it by one less that value, and then add the two. This results of an average of the old information with the new information, the process is illustrated in the following graphs. First we show the long-term profile, together with the new day's data.

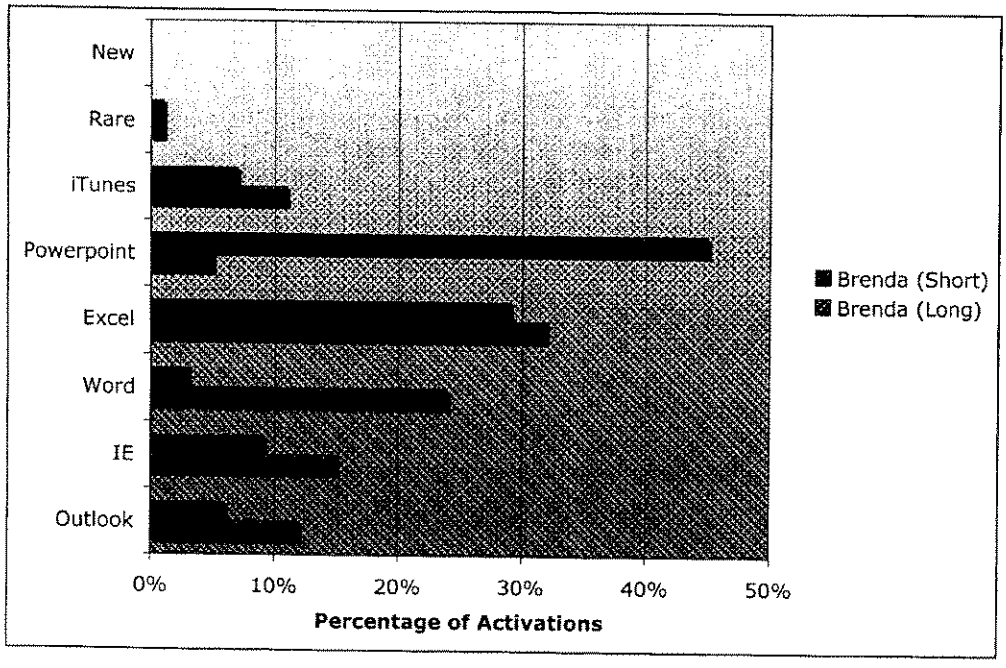

Figure 21: Long term profile for Brenda, together with information at the end of the day ready for incorporation with long-term profile

Next, we show what happens after both profiles have been scaled down. In this example, we multiply the new profile by 1/8, and the old profile by 7/8:

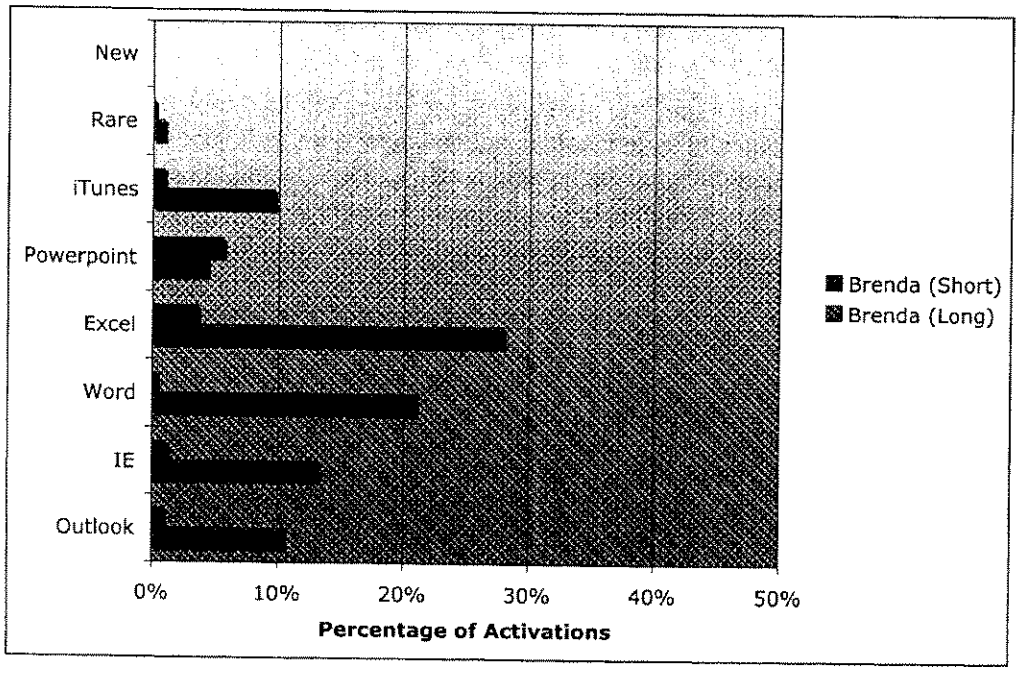

Figure 22: Long term profile for Brenda, together with information at the end of the day ready for incorporation with long-term profile. The long-term has been scaled down by 7/8, and the short-term by 1/8.

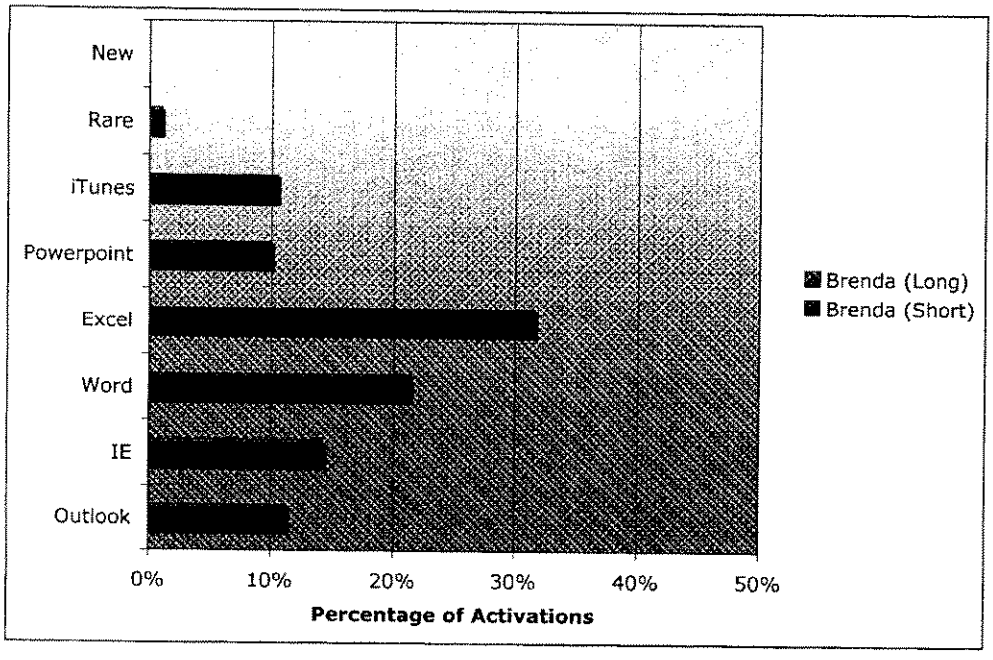

Figure 23: The new long-term profile showing the contribution of the old long-term profile added to the contribution from the most recent day

Note the way in which the very strong representation of PowerPoint in the recent day has had the effect of increasing PowerPoint in the long-term profile. If Brenda, having finished her presentation, did not use PowerPoint again for a few days, the value of PowerPoint in the long-term profile would slowly decay until the next time she worked on a presentation.

If the reader thinks about the influence of any given day in the long-term profile, it is at its greatest immediately after it has been incorporated. The next night, that particular day's contributions will, like all of the long-term profile, get multiplied by 7/8 in order to combine in the latest data. Each subsequent night, another muliplication with 7/8 will occur. After approximately five nights, the day in question will have had it's influence reduced by half. After 10 nights, it will be down to 1/8, and after 15 nights, 1/8. Clearly, days from long ago will have hardly any influence on the current long-term profile. Exponential aging forgets old data, and overweights the newest data. This allows it to adapt to changes in the profile. The time required for the influence of a given contribution to fall to half is known as the *half-life* of the exponential aging process. In this example, we make it about five days for illustration, but the NIDES documents record a half life for the long-term profile of 30 days. Thus it would take NIDES several months to forget things that had happened.

The short-term profile is computed in a basically similar manner, except it is updated after every audit record instead of every day. Thus it responds much more rapidly to changes in the user's behavior. Note that the short-term profile is different than the accumulated days statistics used to update the long term profile in the way NIDES was designed. The function of the short-term profile is be compared against the long-profile to see if the two are becoming very different.

Specifically, the first thing that is computed in differencing the short-term profile and the long-term profile is what NIDES refers to as the *Q-statistic*. The Q-statistic is computed by taking the difference in the heights of the bars of the short-term profile and the long-term profile. Those differences are scaled down by the average variance in the short term profile bars. They are then summed and squared. This gives a metric which will be zero when the short-term profile and the long-term profile are of identical shapes, and will get bigger the more the two have a different shape. The process is illustrated via the following figure.

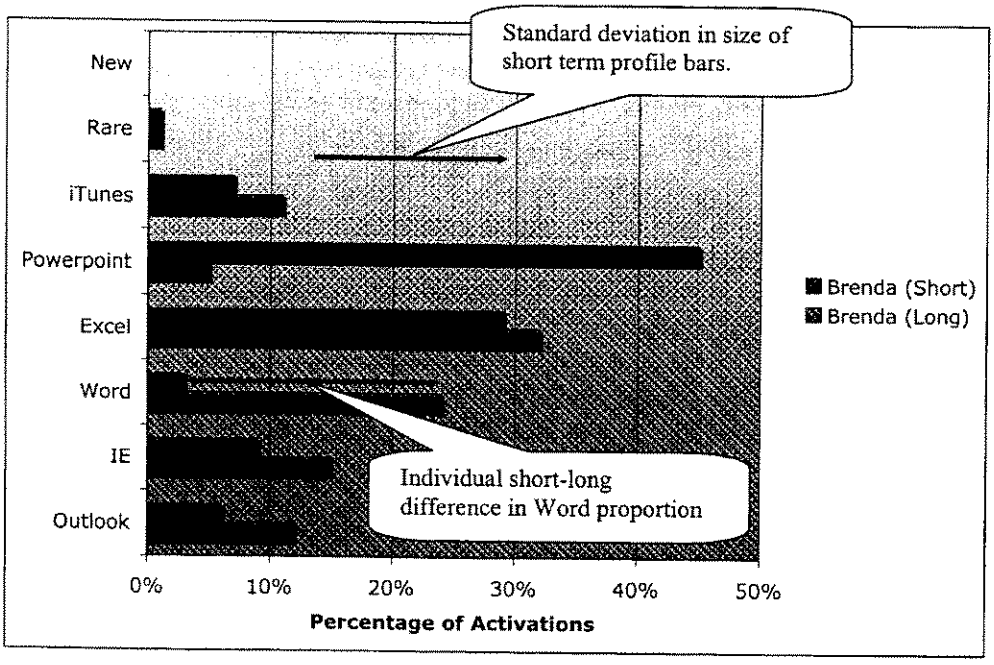

Figure 24: The process of computing the Q statistic of the difference between short term and long term profiles. The individual differences (black arrow) are each computed, and then divided by the variance in the short term profile frequencies (red arrow).

However, NIDES does not use Q directly. To understand why, consider that different users may vary widely in how much their short-term profile fluctuates from the long-term one. Imagine comparing Alice, our retiree earlier, with Carl, who's job is to test new applications before they are deployed on his employer's network. Alice uses almost exclusively two applications, and only the relative frequency of her use of them varies, and that mildly. By contrast, Carl is constantly using new applications, and discarding the use of ones he tested earlier. Carl's profile is constantly shifting and unstable, and his long-term profile is invariably struggling to catch up with his short-term profile from a great distance.

To handle this, NIDES adds an extra layer of transformations onto the Q variable. The first thing they do is maintain a histogram over time of the values the histogram takes on. Thus they keep track of the fact that Alice's Q is usually in a narrow peak close to zero, while Carl's Q is never very close to zero, and is spread out over a large number of variables.

Then they construct a new variable S, which is analogous to Q, - it is big when Q is big and small when Q is small. However, it is explicitly constructed to have the standard normal distribution reviewed earlier. This is a non-parametric technique – Q has an unknown distribution, but by tracking it explicitly in detail via histograms, it is possible to create an S variable which has a well-known and simple distribution.

The final step in the NIDES algorithm is to note that the S for the Application measure is just for a single measure. NIDES and its derivatives were intended to work on many measures at the same time. The way they are combined is simply to take a special kind of average of all the S statistics (the root mean square), and thus create what NIDES calls the T^2 statistic. T^2 effectively summarizes the combined total anomaly of the short-term profiles of all measures compared to their long-term profiles. T^2 by construction has a well-known parametric distribution called the χ^2 distribution. This means it is straightforward to assess whether or not T^2 is anomalous, and how statistically significant the anomaly is, if there is one.

Whether the anomaly has anything to do with a security violation or simply reflects a user changing their behavior is another question again.

The "application usage" measure we just discussed is a type of what NIDES calls a *categorical measure*. This means a measure that discusses the relative frequency with which a variable takes on one of a number of possible distinct values. NIDES considers several other kinds of measures:

- *Intensity measures* look at how often audit records (or events or packets) arrive at the system
- The *audit record distribution* measure is a special case of a categorical measure that explores the distribution of different types of audit records (or events).
- *Counting measures* explore the distribution of a numerical value of some kind (such as the amount of cpu used by a computer, or the number of bytes of data in a connection).

That completes our discussion of the NIDES algorithm. Some detail and some cases have been left out for the sake of brevity, but this description should suffice to illustrate the issues in this case.

IIg) Intrusion Detection Research at UC Davis

The intrusion detection group at UC Davis was founded by Professors Karl Levitt (who retired from SRI and took a faculty position with UC Davis), and Biswanath Mukherjee (a Networking research professor). Together with their students, particularly L. Todd Heberlein, they did some of the most important early research in intrusion detection after SRI.

IIg1) Network Security Monitor (NSM)

The first and most influential system that UC Davis developed was the Network Security Monitor (NSM), developed beginning in the late 1980s, and described in publications from 1990 onwards [Heb90, Heb91a, Heb91b, Heb91c, Muk94, Heb95]. The NSM was the first *Network Intrusion Detection System (NIDS)*. The *Network* in NSM refers to the fact that the system worked by directly inspecting network packets as they were transmitted on a network wire between different computers, rather than inspecting activity on one or more computer systems by looking at their audit trails. All previous systems worked from audit trails.

Operating from network packets turned out to have considerable practical advantages over working from host audit trails, and this style of intrusion detection has proven more important in commercial practice, and spawned many commercial product versions of basically the same idea. The ability to monitor the network allows NIDS systems to be installed at network choke points where the behavior of large numbers of computers can be conveniently monitored from a small number of locations. Additionally, there is no resource impact on the computers being monitored (though the computers or other devices doing the monitoring may need considerable resources). These are practical and administrative advantages that the marketplace has found compelling.

The NSM itself was deployed initially at UC Davis, and the published papers provide a variety of operational experiences there detecting actual intrusion, and some quantitative evaluation of the system. Variants of the same software went on to be widely deployed at US Department of Energy and US military computer installations for the better part of two decades. In the mid to late 1990s, almost the entire US Air Force unclassified network was monitored by ASIM – a descendant of the original NSM. NSM worked exclusively from TCP/IP packets taken from Ethernet networks.

The NSM employed a mixture of statistical and rule based techniques to decide whether or not a given piece of network activity was suspicious. The algorithms were generally inspired by IDES (the original IDES paper was published in 1986), but differed considerably in detail. NSM also discloses several different approaches to the problem that are important to this case, and which I will describe. NSM predates NIDES and thus was not influenced by that later system.

NSM built a warning score that was a weighted sum of three components. There were:

- A statistical anomaly detection algorithm that looked for strange connections and traffic mixes.
- A rule-based system that looked for known bad patterns of traffic.
- A heuristic that assessed the security of endpoints and assessed an intrusion to be more likely if an insecure system was putatively attacking a secure system than the other way around.

The NSM relied heavily on the concepts of *traffic matrices* and *traffic masks*, which I will summarize here. A *matrix* is a mathematical construct that is similar to what is informally known as a table. NSM used a four dimensional matrix, but we shall introduce the concept with a two dimensional simplification, and then work from there.

The dimensions that NSM worked with were the source IP address (of a packet or stream), the destination IP address (of the same), the service (destination port), and a *connection-id* (which was a specially constructed variable designed to ensure that every connection had a different value. Let us start just by working with the source and destination address. Suppose there were four hypothetical computers on the network, which we shall label ".1", ".2", ".3", and ".4" (this denote the different IP addresses of the computers. Let us construct a table with the odds that each of these computers will talk to one another on any given day. It might look like the following:

		Destination			
Source		.1	.2	.3	.4
	.1	0%	100%	96%	68%
	.2	100%	0%	0%	0%
	.3	96%	0%	0%	54%
	.4	68%	0%	54%	0%

Table 2: Traffic mask for hypothetical small network

The way to read this table is that the source labels the rows and the destination labels the columns. The percentage in each cell is the probability (odds) that a given source will contact a given destination on a given day. Thus, for example, this traffic mask estimates there is a 68% chance that the computer with address ".4" will source packets to ".1`" on a randomly chosen day (the lower left cell in the table.. In this example, all the data-flows are symmetric so the upper right cell, the probability that ".1" will source packets to ".4" is also 68%. The diagonal elements of the table are the probabilities that hosts will send packets to themselves. Since these would not appear on the network but be delivered internally, those probabilities are zero. In our example, all the computers talk to .1 with some frequency, but other than that, only .3 and .4 talk. In realistic examples on larger networks, only a small proportion of the possible conversations actually have significant probability of occurring – most computers do not talk to most other computers (just as in a large office, most people do not talk to most other people but only to a smaller circle of coworkers.

Thus if on some occasion .2 suddenly began talking to .4, we could say that this was anomalous. Of course in a complex and changing network, a single anomaly like this might well just represent an innocent change in behavior. However, a large number of anomalies would be likely to represent either an intrusion, or some kind of network or computer problem causing network traffic to be unusual. This was the basic idea behind the statistical approach of the NSM.

Now, to get closer to what the NSM was doing, we need to complicate the picture more. If the reader will imagine taking the two dimensional table above and stacking many instances of them on top of each of other to gain a three dimensional table, then each layer in that three dimensional table could be indexed by the service (or destination port, such as for the world wide web (HTTP), email (SMTP), and so on. Thus each cell in the three dimensional matrix corresponds to the possibility of connections from a particular source to a particular service on a particular destination. Now if the reader imagines taking a large stack of these three dimensional tables and spreading them out (along some admittedly hard-to-visualize fourth dimension), where each cell in the fourth dimension corresponds to different occasions when there was a connection between that source and destination on that particular service (there could be many such connections over the course of the hours and days of operation of the network), then taking each cell in the

four dimensional matrix and storing the number of packets and bytes associated with that direction of the connection, then the reader will have some idea of what the NSM meant by a *traffic matrix*.

A *traffic mask* was similar, except that instead of storing information about each connection that occurred, it stored a statistical description of the probability of a connection occurring for a given source, destination, and service combination, and it also stored an estimate of the probability distribution of a given number of packets being in the connection, and a given number of bytes being transmitted in the connection.

Every five minutes, the traffic matrix of current traffic was compared to the traffic mask, and the anomalousness of the matrix was computed. Every 15 minutes the current traffic mask was saved to disk. At long intervals, the traffic mask of normal traffic was computed from a large set of the current traffic matrices that had been stored every 15 minutes.

The probabilities of connections occurring were estimated using the same exponential aging approach described in the NIDES algorithm [Heb91a].

In addition to the statistical approach, NSM also implemented some fixed rules including

- Sources connecting to more than 15 destinations were suspicious
- Sources trying to login more than 15 times were suspicious
- Sources trying to connect to non-existent destinations were suspicious.

IIg2) Distributed Intrusion Detection System (DIDS)

After the NSM, the next major system that UC Davis developed was DIDS. DIDS was deployed at UC Davis, and subsequently was re-engineered by Trident Data Systems and deployed at least at test sites at the US Air Force. The UC Davis version of DIDS, described in a series of publications [DIDS, DIDS-Feb, DRA, SNA 91] was a prototype intended to research the issues with integrating and correlating information from a variety of sources in a computer network to come up with a coherent single picture of distributed intrusive activity. DIDS was the first system to use the architecture that has since become widespread commercial practice in the industry in which a set of sensors examine data from hosts and networks, perform some detection and data reduction locally, and then feed activity to a centralized management platform which does further correlation and detection and presents a coherent picture to users of the system.

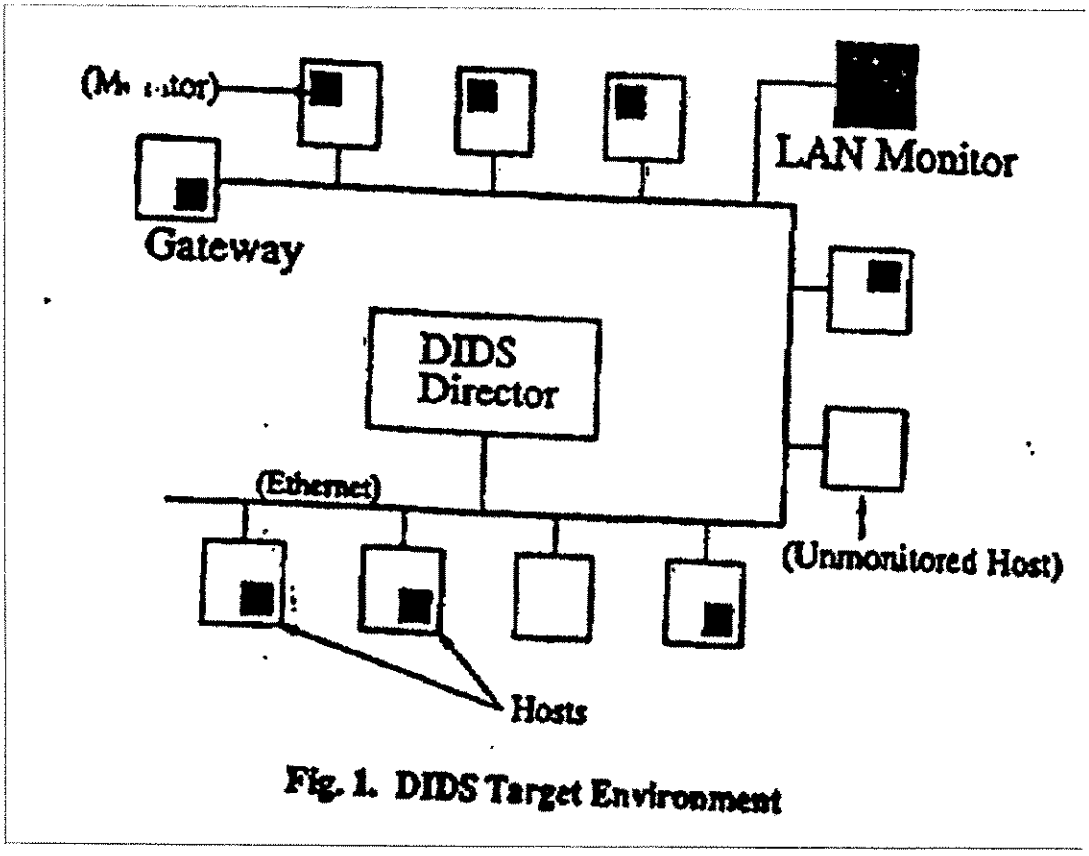

Figure 25: (Figure 1 of [DIDS]) showing an overview of DIDS deployed on a computer network.

Figure 1 shows the hosts (square hollow boxes) attached to a network (lines). The black boxes inside the hosts represent the DIDS *host monitors* that ran on each host, examined the host audit trail, and created an event stream of significant happenings or potential intrusions. The large black square at top right of the figure is the LAN Monitor. This was responsible for watching the network looking for intrusions and creating an event stream tracking activities on the network required for correlation. The DIDS LAN Monitor was based on the NSM (described above). Finally, the large rectangle in the middle of Figure 25 is the DIDS Director. This was the centralized component that took events from the LAN Monitors and Host Monitors, drew overall conclusions, and presented them to the user.

The flow of communication is shown more explicitly in Figure 26:

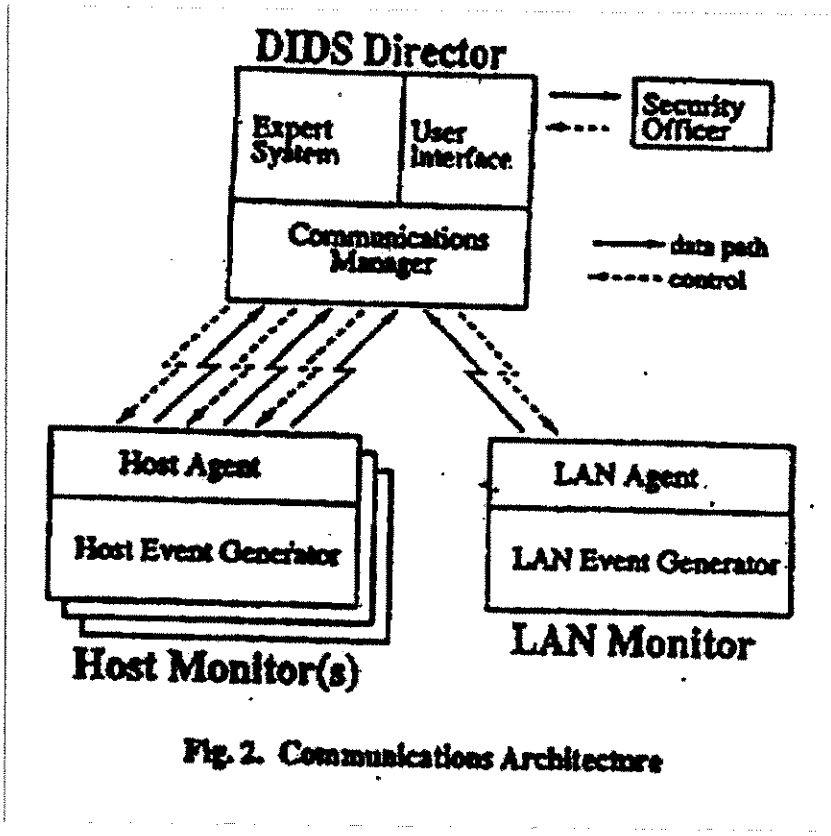

Fig. 2. Communications Architecture

Figure 26: (Figure 2 of [DIDS]) showing communication between the elements of the DIDS system.

The architecture shows the Host Monitor split into a Host Event Generator which reads the system audit trail, applies signatures to it, and creates DIDS event records as needed, and the Host Agent which is responsible for communication with the DIDS director. The LAN Monitor has a similar architecture – a LAN Event Generator, which is basically an NSM, together with a LAN agent which is responsible for interacting with the Director. The Director itself consists of a communication portion which interacts with the various monitors, the Expert System component that did actual correlation and reasoning, and then a user interface which handled communication with the human users of DIDS.

The DIDS Director combined events from the LAN monitors and the Host monitors. It did this based on commonalities amongst the events being reported to it. For example, extensive effort was expended [DRA] to ensure that users were tracked as they moved from host to host so that all their activities could be accounted to a single *Network User ID (NID).*

IIg3) Graph-Based Intrusion Detection System (GrIDS)

The GrIDS project at UC Davis began in late 1995, published a first paper in late 1996, by which time the system was essentially implemented, and continued for several more years with diminishing levels of effort. Most descriptions in this report are as of a technical report which described the design of the system in some detail and a version of which was published on the UC Davis website no later than August 27th, 1996. The author of this expert report was also the technical leader of the GrIDS effort.

The GrIDS team worked collaboratively to design the system through 1996. The GrIDS technical report was begun in February of 1996, and largely completed by May of 1996. At some point no later than August 27th 1996, when I gave a presentation about the system to a DARPA PI meeting in Santa Cruz, CA, I placed it on the GrIDS web site (part of the UC Davis computer security laboratory website) where it was available to the world. It was then updated at regular intervals.

The motivation for GrIDS was to push beyond the then state-of-the-art in intrusion detection, and the central direction of that push was to be able to perform intrusion detection across large networks – either the network of a large organization, or perhaps eventually even the entire Internet. The goal was partly to organize, correlate, and display the alerts from individual sensors (network intrusion detection systems or host intrusion detection systems for individual computers), but more importantly, to detect large scale attacks on the network, which might only be detectable by correlating information from multiple locations. The system was especially targeted at scans and worms, but also could put together chains of actions by individual hackers compromising a number of machines in sequence.

GrIDS put together activities into *GrIDS graphs*, an example of which is shown here. I created this picture and presented it at as part of the slideshow presentation at a DARPA Principal Investigators meeting at Santa Cruz in 1996. It is an actual screen shot from a running system in August 1996.

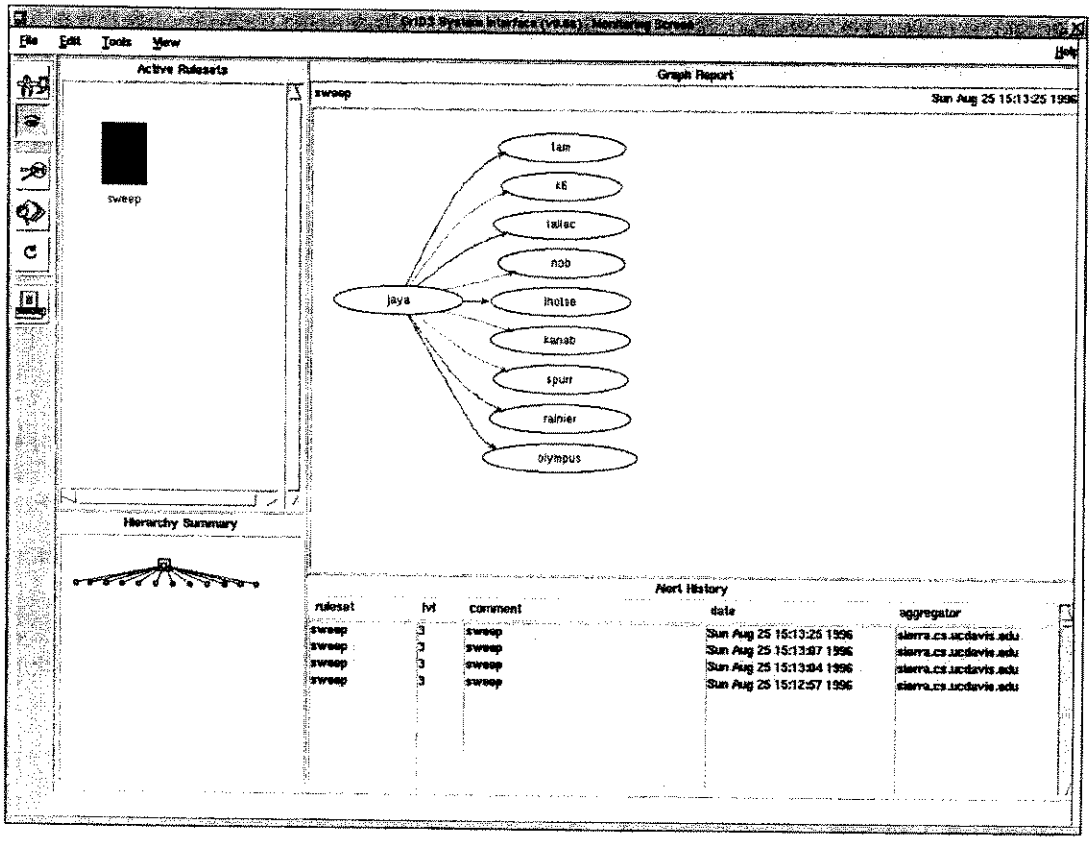

Figure 27: A screenshot of the main monitoring screen in the GrIDS user interface

The large top right panel in the user interface shows a graph of the kind that GrIDS was trying to detect and visualize. A graph is a kind of diagram used by computer scientists to understand a variety of phenomena. Graphs have *nodes* (the labeled ovals in the picture), and *edges* (the lines which join the nodes). In GrIDS, the nodes were always hosts on a network, and the edges represented that some kind of network traffic had been made between those hosts. Typically this would be a TCP connection, but not necessarily. The assertion made by a graph that has been reported to the user interface like this is essentially "this pattern of activity on and between these hosts is all causally related, and it is suspicious". The formation of these graphs was the way GrIDS correlated data from a variety of data sources on the network.

In the particular case shown, the activity is a sweep. The computer named "Jaya" has connected to 9 other computers in a manner that is sufficiently rapid to have caused GrIDS to detect it and label it suspicious.

GrIDS was designed to be a system that could take input from a variety of sources of data. These could be other intrusion detection systems reporting problems on a particular host, filters that translated host logs into GrIDS data, or network systems reporting connections or various kinds of network activity, A special language was defined for

expressing all this data in GrIDS graphs. Data that was about a particular host would form a *node report*, while data that was about some kind of network activity between two hosts would form an edge report.

One particular source of data was a network sniffer component that essentially translated packets on the network into GrIDS graph reports that were then consumed by the local graph engine. For example, it would create records when a new network connection was requested, not whether it succeeded or was denied, and so forth. These would become edge reports that went into the lowest level graph engine.

Note the "Hierarchy Summary" panel in the lower left of the screenshot in Figure XXX. An important aspect of GrIDS is that all prior distributed intrusion detection systems (eg DIDS or NIDES) organized processing only on the sensor where the data was initially generated (either from host audit trails or packets), or on a centralized platform that was responsible for all correlation reasoning: reasoning where data from several sensors had to be combined to draw a conclusion. In GrIDS, correlation and inference could be performed across a larger hierarchy of processing modules (called *graph engines* in GrIDS). The idea is illustrated with the following diagram (again taken from the 1996 presentation).

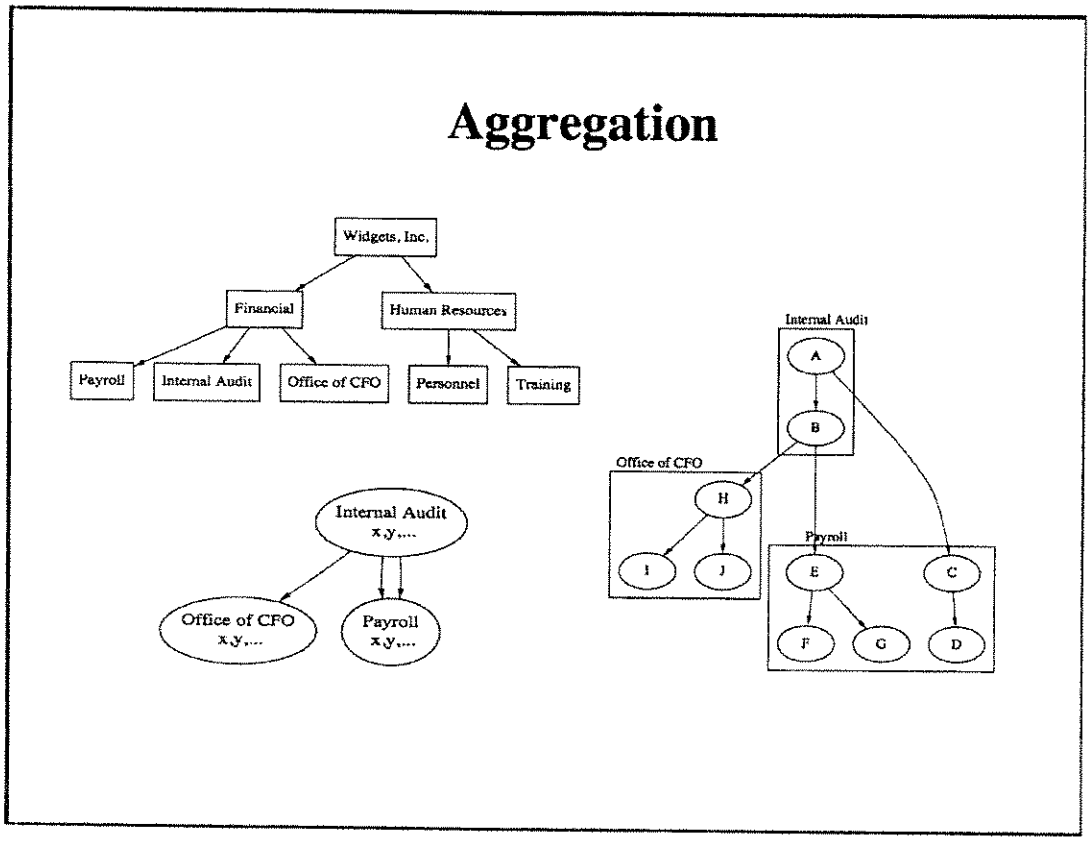

Figure 28: Aggregation of graphs in the GrIDS hierarchy

In the top left, we see part of the organizational chart for a hypothetical enterprise (Widgets, Inc) on which GrIDS might be deployed. The organization is decomposed into a hierarchy of *departments* with sub-departments inside the departments. Thus for example, the "Payroll" department is part of the "Financial" department within Widgets, Inc. These departments are assumed to contain computers (and any computer is in exactly one department). Each department had its own graph engine, which was responsible for doing correlation of activity within that department by forming graphs of suspicious activity. This particular hierarchy has three levels of departments, but GrIDS could have arbitrarily many such levels.

A department might contain a variety of data sources, but only one GrIDS engine to which all the local data-sources would send their reports in the graph language. The graph engine would combine them into graphs.

The graph engine itself was fairly generic. It contained code for maintaining the graph structures, interpreting the graph language, and deciding whether two graphs overlapped (which was a necessary condition for them to get combined). However, the detailed logic for deciding exactly how to build graphs was contained in *rulesets*. GrIDS could run a number of rulesets at the same time, and they would separately build different kinds of graphs according to their own logic. Graphs from one ruleset would never interact or combine with graphs from another ruleset.

The rulesets were written in a rule language that specified under what circumstances nodes and edges would combine into graphs, and graphs combine into larger graphs. Two graphs could only combine if they overlapped in their endpoints (that is they must have at least one host in common). Beyond this, the ruleset could make fairly arbitrary decisions based on the attributes of the nodes and edges in the potentially combining graphs. For an example, it was common for graphs to only combine if the time attribute at nodes where close enough. Thus if two edge reports that might be parts of a possible scan occurred three seconds apart, they could combine. If they occurred two days apart, they couldn't. Indeed graphs timed out after a while, so the older one might not be available to combine.

The ruleset also specified when graphs were suspicious enough to be reported to a user interface. Generally, graphs were decided to be suspicious either if the graph was large (which was typical of worms and scans), or a data-source had labeled one of the nodes or edges as suspicious directly (in which case the entire graph would inherit that suspicion). However, in principle the ruleset allowed the decision about whether a graph was suspicious or not to be made based on arbitrary properties of the graph. Graphs that cut across multiple departments were always passed up to the graph engine of higher departments in the hierarchy, but graphs were also passed up the hierarchy if the ruleset changed a global variable in the graph (which would invariably be the case if the graph had crossed the threshold to be alerted to the user interface, but could also be done at the ruleset's option for other reasons of correlation).

In the lower right of the figure, a worm graph is shown as the worm spreads through the computers in several departments. The worm has started on computer A in the Internal Audit department, and then spread to computer B in the same department, but also to C in

the Payroll department. From B it separately spreads to E in Payroll, and also to H in the Office of the CFO. And so forth.

If all the computers infected by the worm were in the same department, GrIDS would be straightforwardly able to build the whole graph, notice that it was large, and decide it was suspicious and worm-like. However, because the graph is split between multiple bottom level departments, each of them cannot see the whole picture. Indeed the graph engine for the Payroll department cannot even tell that the CD graph and the EFG graph are related rather than separate, since that connection occurred in the Internal Audit department.

The way GrIDS solves this problem is that all graphs that cross a department boundary are passed up to the next higher department in the hierarchy (in this case Financial). However, to make the system more scalable, GrIDS did not pass up the entire graph (which could be very large in a realistic system), but rather just a reduced graph in which the department appeared as a single node. Special attributes of this reduced graph carried information like the number of nodes (hosts) and edges in the graph. This allowed the Financial department to build the graph shown in the lower left of Figure 28. However, crucially, the attributes of all the graphs could be added together and thus the Financial department graph engine could tell that the graph was in fact large enough to cross the threshold and be considered suspicious.

This last figure displays an actual three level hierarchy running on notional departments within the computer security lab at UC Davis in 1996. This again is from a slide in the presentation to the Santa Cruz PI meeting [Santa Cruz PI].

All of the above was described in detail in the May 1997 Technical Report available on the web. [GrIDS].

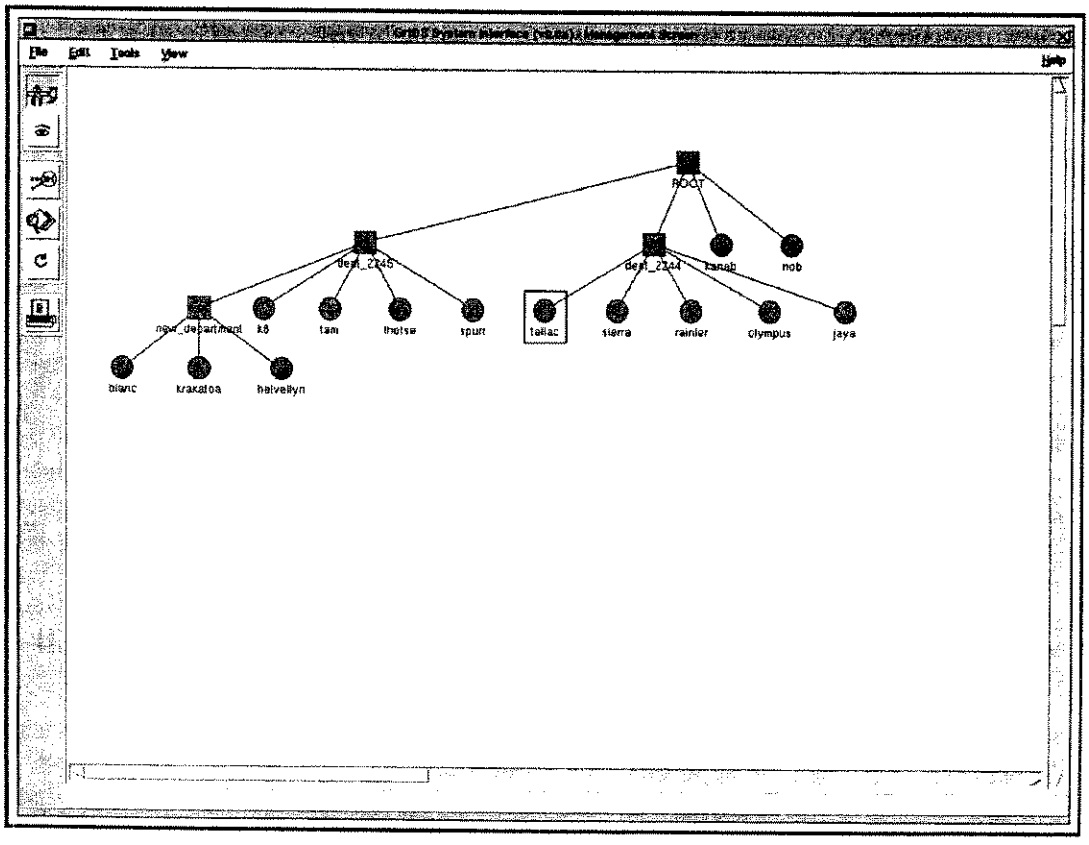

Figure 29: A screenshot of a running GrIDS user interface showing the hierarchy available for management.

IIi) DARPA PI meetings and CIDF

It is important to understand that from 1996 on, the Defense Advanced Research Projects Agency, under the auspices of Program Manager Teresa Lunt, began a substantial ramp up of funding into intrusion detection research. This led to the formation of a community of researchers who regularly attended DARPA PI meeting. The researchers working on Ms Lunt's program who regularly attended PI meetings included at a minimum Karl Levitt and myself from UC Davis (GrIDS), Phil Porras and Al Valdes from SRI (EMERALD), Felix Wu of NCSU and Frank Jou of MCNC (JiNao), Dan Schnackenberg of Boeing (IDIP), Brian Tung of ISI (CRISIS), Maureen Stillman of Oddyssey Research, Rich Feiertag of Trusted Information Systems, and Cliff Kahn of the Open Group, as well as numerous others. We were required to disclose our technical progress to one another at PI meetings (including in Santa Cruz, CA in August 1996, and in Savannah, GA in February 1997)

Additionally, Teresa Lunt, our Program Manager required our projects to work together to develop the Common Intrusion Detection Framework (CIDF), an effort to develop

common APIs and means for intrusion detection systems to interoperate with each other. A short history of this effort is available from [CIDF]

> "The goal of the Common Intrusion Detection Framework is a set of specifications which allow
>
> - "different intrusion detection systems to inter-operate and share information as richly as possible,
> - "components of intrusion detection systems to be easily re-used in contexts different from those they were designed for.
>
> "The CIDF working group came together originally in January 1997 at the behest of Teresa Lunt at DARPA in order to develop standards to accomplish the goals outlined in the previous section. She was particularly concerned that the various intrusion detection efforts she was funding be usable and reusable together and have lasting value to customers of intrusion detection systems.
>
> "During the life of the effort, it became clear that this was of wider value than just to DARPA contractors, and the group was broadened to include representatives from a number of government, commercial, and academic organizations. After the first few months, membership in the CIDF working group was open to any individuals or organizations that wished to contribute. No cost was involved (except to defray meeting expenses).
>
> "Major decisions were made at regular (every few months) meetings of the working group. Those decisions were made by rough consensus of all attendees. That is, the meeting facilitator attempted to reach consensus, but in situations where only one or two individuals were protesting a decision, they were overruled in the interest of efficiency. No decisions were taken in the face of opposition from a sizeable minority, rather the issue was tabled for further consideration. Meetings were fun and the working group had a good time doing this (well, most of them, anyway).
>
> "In between meetings, most of the writing was done by small subgroups or individuals. Their text was brought back for approval/changes at meetings. Discussions were also carried on in the working group mailing list, but few decisions were made that way.

An overview of the architecture of the resulting system was also in that document:

All CIDF components deal in *gidos* (generalized intrusion detection objects) which are represented via a standard common format. Gidos are data that is moved around in the intrusion detection system. Gidos can represent events that occurred in the system, analysis of those events, prescriptions to be carried out, or queries about events.

CIDF defines four interfaces that CIDF components may implement:

	Push-style	Pull-style
Producer	Produces gidos when it wants to, typically in response to events.	Produces gidos when queried.
Consumer	Mates with push-style producer.	Mates with pull-style producer.

Each of these interfaces takes two forms: a callable form, which permits reuse of the component, and a protocol form, which permits the component to interoperate with other CIDF components.

CIDF defines several types of preferred components:

* Event generators
* Analyzers
* Databases
* Response units

Figure 1.1 presents a schematic view of these components in a hypothetical intrusion detection system. The solid boxes labeled E1, E2, A1, A2, D, etc represent the various components of some hypothetical intrusion detection system. It is convenient to think of these as objects in the object-oriented programming sense (this does not dictate an implementation in an object-oriented language or framework).

Phillip Porras and I were two of the coauthors of that document and the CIDF group had been working together on that architecture and those interfaces for ten months before the statutory bar date, and 20 months before he filed the application that became the 338 patent.

In my opinion, the CIDF standard as it emerged had large overlaps with the patent claims filed by SRI. At no time did Mr Porras reveal to me that he was filing patents covering broad areas of our joint work on CIDF.

IIj) Emerald

In 1996 SRI resumed intrusion detection work under new leadership. With Teresa Lunt seconded to DARPA, Phillip Porras came on board to head the intrusion detection group, working with veterans of IDES and NIDES including Peter Neumann and Al Valdes. It was during the Emerald project that the patents in suit were to be filed.

EMERALD uses the NIDES techniques: a statistical inference component using the NIDES algorithm (described above), a signature/rule based inference engine for looking for patterns in the data indicative of known misuse, and a resolver to take the outputs of those other components and decide what to do.

The main focus of Emerald was to take the same techniques SRI had already worked on, and perform a reengineering of them in a way that was intended to be more scalable, more flexible for deployment in large networks, and to work on more different kinds of data. This vision has not had any discernable impact on commercial intrusion detection systems, which instead have evolved in the direction of prepackaging the software with hardware as appliances which can be dropped into the network, and have largely stuck with the architecture (pioneered in DIDS) of having a set of fixed sensors feeding a central correlation server. This latter architecture has continued to win out in practice because it is simpler to administer, and trained humans are the things in shortest supply in most IT departments, so simplicity of management is a key driver of technology adoption. Staff costs for management are typically the largest part of the total cost of ownership (TCO) of an intrusion detection system. Complex schemes such as Emerald that require a lot of configuration tend to push staffing costs up.

IIk) Patent Claim Analysis

The four patents in suit are all related because the latter three are continuations of the first. The patents and their relationship are briefly as follows:

- Patent #6,321,338 (the "338 patent") was initially filed on November 9th, 1998, and issued on November 20th 2001. It was the first patent SRI filed in the area of intrusion detection.
- Patent #6,484,203 (the "203 patent") was filed on September 8th, 2000 and issued on November 19th 2002. It is a continuation of the 338 patent, which means that it has the same priority date, and shares the exact same specification of the invention, but has a different set of claims that SRI came to assert, and the patent office to accept.
- Patent #6,711,615 (the "615 patent") was filed on September 25th 2002, and issued on March 23rd 2004. It is a continuation of the 203 patent (itself a continuation of the 338 patent).
- Patent #6,708,212 (the "212 patent") was filed on May 5th 2003, and issued on March 16th 2004. It is also a continuation of the 203 patent.

All four patents, when they reached the US Patent Office, were reviewed by Mr. Thomas M. Heckler as Primary Examiner. The continuation structure of the patents can be illustrated with the following diagram.

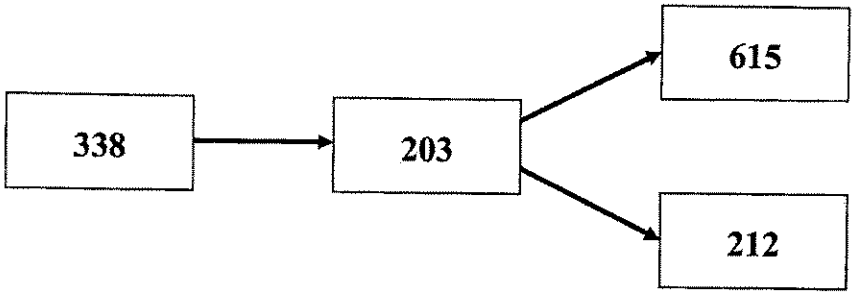

Figure 30: Continuation Structure of the Patents in Suit

All four patents contain the same detailed description and figures.

Let us now turn to the patent specification as described in the figures and the detailed description of those figures.

I understand that the patent specification and claims are to be understood as they would be understood by one of average skill in the art. In my opinion, a person of average skill in the intrusion detection field would either have (a) a PhD in computer science working on intrusion detection or (b) a BS in computer science and two years experience implementing an intrusion detection system.

Summary of Specification

In this section, I will provide an overview of the patent description organized around the figures.

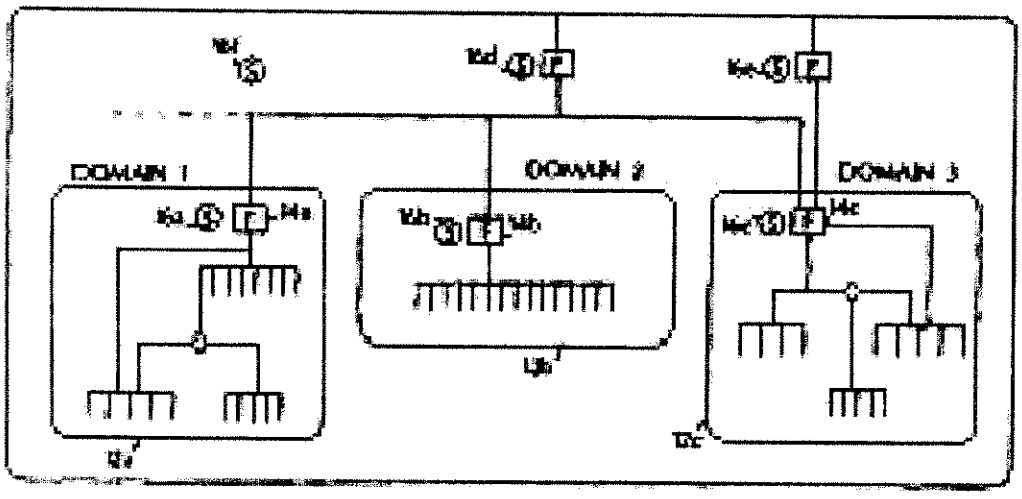

FIG. 1

Figure 31: Figure 1 of the patents, showing deployment of Emerald monitors in an organization. Additional labels have been added in blue for improved clarity

Figure 31, which is identical to Figure 1 of the patents (with the addition of the labels) illustrates the deployment of EMERALD monitors throughout the enterprise. As you can see, a series of service monitors are deployed for the sake of monitoring individual network domains via network traffic. These in turn send their conclusions to the domain monitors which perform analysis on those conclusions. In turn, the domain monitors send their outputs to the enterprise monitor which analyze that data.

All monitors have the same software architecture and use the same reconfigurable monitor code. This is illustrated in Figure 32.

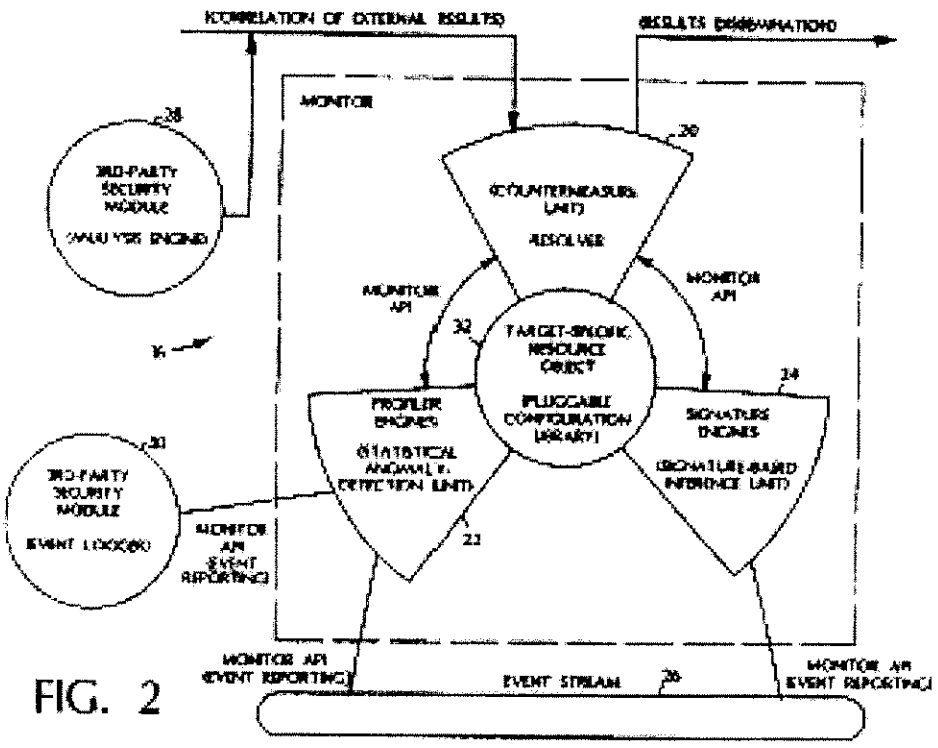

FIG. 2

Figure 32: Figure 2 of the patent specification summarizing the software architecture of an EMERALD monitor

The conical sections around the central circle are the main analysis and decision-making engines: the statistical anomaly detection engine on the lower left, the signature engine on the lower-right, and the resolver at the top. These three components are always identical in code in every monitor. What distinguishes the different kinds of monitor is the configuration resource object shown as the circle at the center of the picture. This tells the other components what kinds of event format to expect, what rules/measures/response policies to apply, and in general supplies all site-specific or updatable configuration.

The diagram also shows the interfaces to the monitor around the edges – events coming in through a standard interface at the bottom, results being sent to other monitors at top right, or coming from other monitors at top left, and opportunities for other systems to provide events via APIs at the left.

Figure 33 summarizes the contents of a resource object.

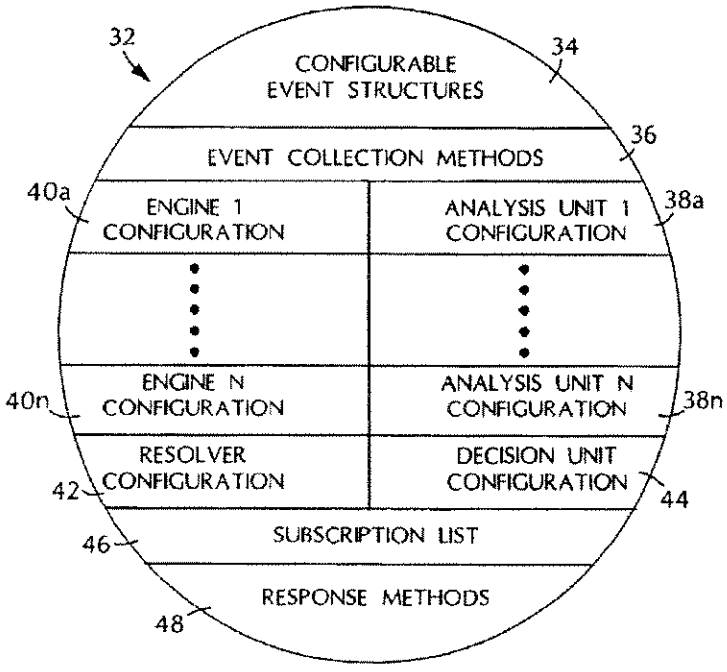

FIG. 3

Figure 33: Figure 3 of the patents summarizing the contents of a monitor resource object

Starting from the top, we have the event structure configuration which is supposed to tell the monitor the structure of its event stream, allowing the correct fields to be incorporated into the correct statistical profiles, signatures, etc. Below that are methods (ie code) for parsing the event stream for this particular kind of monitor. Then come a series of configuration parameter objects to define the behavior of each statistical profile unit, and each signature-based unit. (Note that there can potentially be more than one of each kind per monitor). Below these are the configurations for the resolver (which tell it how to interpret the outputs of the analysis engines and what responses to take in consequence).

Moving down to the bottom, we have the subscription list of which other monitors this monitor should communicate with, and then at the very bottom we have code methods to implement responses.

It is worth noting that no syntactic or semantic detail for any of these resource object components is disclosed in the patent specification. Essentially all the patent has is high-level functional requirements for these objects.

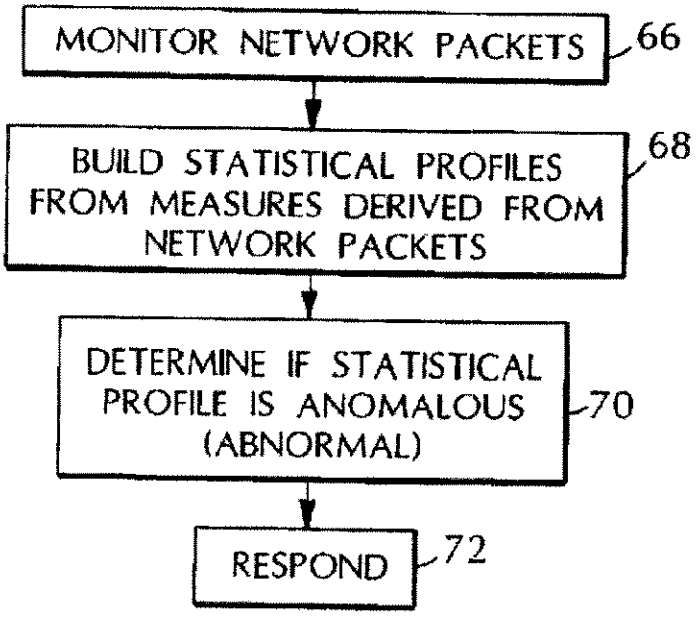

FIG. 4

Figure 34: Figure 4 of the patents showing the process of building and comparing statistical profiles and deciding whether to respond

Figure 4 simply covers the idea of taking network packets, building profiles, determining if the profile is anomalous, and then taking a response. Figure 5 covers the similar idea for event records, but where one of a selection of short-term profiles must be selected for the event to be added to before comparison with a long term profile.

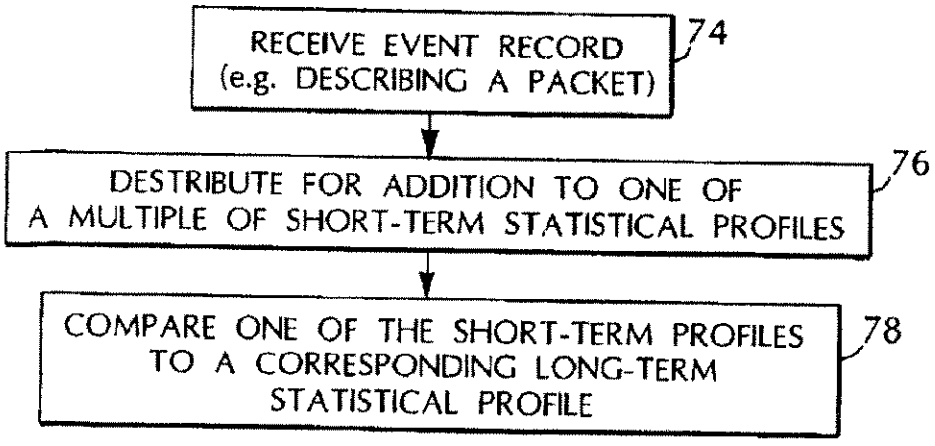

FIG. 5

Figure 35: Figure 5 of the patents showing the process of building multiple short term profiles and comparing to a long-term profile.

Figure 6 is not reproduced here but shows the network monitor instructions on a hard-drive of a computer.

I now turn to a discussion of the validity of the patents in light of certain prior art references.

The 338 patent (Patent No. 6,321,338) – NSM invalidation

It is my opinion that the claims of the '338 patent discussed below are obvious in light of the NSM reference [Heb90], the IDES reference [Ref. 15 of Heb90] and general practice in light of leading commercial systems such as ISS RealSecure [RealSecure 1.0] and Wheelgroup NetRanger [NetRanger]. The NSM reference specifically suggests using masks in a manner similar to IDES profiles.

Except where otherwise indicated, references to "NSM" refer to [Heb90].

Independent claim 1 of the 338 patent says:

> *1. A method of network surveillance, comprising:*
> *receiving network packets handled by a network entity;*
> *building at least one long-term and at least one short-term statistical profile from at least one measure of the network packets, the at least one measure monitoring data transfers, errors, or network connections.*

comparing at least one long-term and at least one short-term statistical profile; and

determining whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

Now, it is clear from earlier description that the NSM conducted network surveillance by receiving network packets handled by a network entity and that is disclosed in the [ref NSM]. It is clear that the NSM was looking at network connections – they were a central part of its data structures.

NSM also builds a long-term statistical profile. As we discussed earlier, the NSM built a "current traffic matrix" and compared it to a "normal/abnormal traffic mask". It is worth quoting several key sections of section 5.2 of [Heb90] that describe the comparison process.

> "Detecting unusual patterns by a probabilistic analysis of the traffic requires knowledge of what the normal traffic flow is. The current traffic matrix is then compared to the normal/abnormal traffic mask to determine if something unusual is happening."

Also,

> "The prototype examines the current network traffic when a new node is added, and at five minute intervals. When a new node is added to the network, the probabilistic analysis examines the cell against the normal/abnormal mask. At five minute intervals, the entire traffic matrix is compared to the normal/abnormal mask and the rules."

And

> "Finally, the NSM's normal profile does not consist simply of a mean and variance; it consists of a range of values and the probability of observing a value at each range. Careful examination of network traffic showed that data amounts were not always Gaussian distributed; therefore, the mean and variance could not capture the true shape of the data."

The construction of the "normal/abnormal traffic mask" is covered in section 5.1 of [Heb90]:

> "Finally, the matrix archiver writes the matrix representing the current traffic out to disk. Currently, a signal to save the matrix is sent to the archiver by the matrix analyzer every fifteen minutes… The archived files can be used to build or update a network profile."

It seems clear that under any reasonable definition, the "normal/abnormal traffic mask" is a "long-term statistical profile". It contains probabilities of what kinds of connections occur, and statistical distributions of how much data is in them – even storing the distributions in a non-parametric manner as in NIDES and later versions of IDES. It is built out of a series of short-term sets of observations of network traffic (the current traffic matrices that are archived every fifteen minutes). This alone amply discloses a long term statistical profile. Other references disclose the same point. Indeed, in a

different NSM reference [Heb91a], we are told specifically that the mask is built using the same kind of exponential averaging that IDES and NIDES used:

> "The probability of observing a connection on the data path is based on whether a connection was observed on the data path on each of the 'n' previous days. This function, taken from [15], is defined as:
>
> $$P_r = \frac{\sum_{k\geq 1}^{n} W_k 2^{-bk}}{\sum_{k\geq 1}^{n} 2^{-k}}$$
>
> where
>
> - k is the index of days. k = 1 is the most recent day, and k=n is the oldest day for which data is known.
> - W_k is an indicator function that returns 1 if there was a connection on the data path for the k^{th} day, and it returns 0 otherwise.
> - b is the half-life of the data paths. For the NSM implementation, I used a half-life of seven days (a single week), so b=1/7=0.143"

In the context of the quoted paper, reference [15] was an SRI interim progress report for IDES dated in 1990. A half-life of seven days certainly qualify as "long-term" relative to the current traffic archive interval of 15 minutes.

The "current traffic matrix" as disclosed in NSM stored records of every connection in the current traffic, rather than a statistical summary of them. However, NSM **could** very easily have built a short-term profile – since it already contained the code to combine current traffic matrices into long-term profiles, it could as well have combined them into short-term profiles – the only real difference would have been to use a short lifetime. Obviously, the technique of comparing long-term and short-term profiles was well-known to the NSM inventors, both because they explicitly cite it, and also just because IDES was far and away the most famous intrusion detection project in the early years of the field and it was the project where that particular idea was first invented. And the NSM specifically mentions generating masks used by IDES. On page 299 of [Heb90] we find:

> "The actual representation of the mask can be performed by several methods. We currently generate a probabilistic distribution of values for each measurement which is seen frequently. The value of a current measurement is compared to the distribution, and if the probability of that value occurring is too low, it is passed through the mask. Although we have had excellent results with this method, it is very expensive in terms of memory requirement, and it may not be appropriate in all environments. We are currently looking at generating masks used by IDES, Wisdom and Sense, and other intrusion detection systems, so that we can make an experimental analysis of these different methods."

So we see the NSM inventors not only suggesting, but investigating, making additional borrowings from IDES as needed, and of course a closer application of IDES to the mask analysis would have led to using short-term profiles and comparing them to longer-term profiles. In my opinion, it would have made little difference either way if they had computed a short-term profile from the current traffic matrix and then compared that to the long-term profile, or instead compared the current traffic matrix directly to the long-term profile. Either way, the amount of work is basically proportional to the cost of traversing the current traffic matrix, and the amount of memory required is dominated by the size of the current traffic matrix.

Thus in my opinion, it would have been obvious in light of IDES to provide a short term profile to NSM. Moreover, the short-term/long-term profile technique was documented in both IDES and NIDES, and the general idea of applying it to network traffic was disclosed in [And95]. Therefore claim 1 is invalid.

Given that, I now consider the degree to which the dependent claim systems would fall under the NSM.

> *2. The method of claim 1, wherein the measure monitors data transfers by monitoring network packet data transfer commands.*

Monitoring network packet data transfer commands was something widely done for a variety of purposes in the industry. As evidence for this, I examined signatures for the two leading commercial products at the time, ISS RealSecure [RealSecure 1.0, 1.1, 1.2 & 1.2.2], and Wheelgroup NetRanger [NetRanger]. This revealed a number of signatures associated with checking data transfer commands of various kinds in the signatures sets. Therefore, this is an obvious extension of any network intrusion detection system, including NSM.

Some examples from [RealSecure 1.0, 1.1, 1.2 & 1.2.2] include:

> "RealSecure can monitor many network services, including … file transfer …" [RealSecure 1.2.2 Chapter 1, p. 1].
>
> **"FTP GET File Decoding**
>
> "Files being transferred from the source host to the destination host use a GET command in order to transfer the files. FTP GET decoding discovers all files that are being transferred to the source host over FTP." [RTARR, p. 4.]
>
> [RealSecure 1.1 pp. A-10 to A-11]
>
> **"FTP PUT File Decoding**
>
> "Files being transferred from the source host to the destination host use a PUT command in order to transfer the files. FTP PUT decoding discovers all files that are being transferred to the destination host over FTP." [RTARR p. 4.]
>
> [RTARR, p. A-11]

These ISS signatures were explicitly monitoring various kinds of FTP file commands.

> **"HTTP GET Decoding**
>
> Pages, images, and all other information that is viewed through a Web browser on the World Wide Web are transferred through HTTP using the GET command. HTTP GET decoding discovers all Web pages that are being transmitted unsecurely to a machine. This allows an administrator to track, log, and view Web traffic on the network." [RTARR p. 4.]
>
> [RTARR p. A-12]

This ISS signature is tracking data transfer commands in the form of HTTP methods within HTTP.

NetRanger had similar facilities, as demonstrated by the following excerpts from [ref NetRanger]. The first monitors data transfer commands in FTP.

> "The following string detects users attempting to FTP the password file. … The following strings are used to detect the username and passwords of users in ftp sessions. These also indicate when directories are created/deleted and when files are GET/PUT." [NetRanger p. 4-79].

Next we have a signature for monitoring data transfer commands in HTTP;

> "The following example shows how to log every HTTP GET on a network." [NetRanger p. 4-80].

Now we have signatures for monitoring data transfer commands in DNS:

> **"DNS HINFO Requests.** This attack uses DNS to gather information about a specific host.
>
> **"DNS Zone Transfer Request.** This attack attempts to gather information about all hosts registered with your DNS server. This can be used by hackers to try to get a map of your network.
>
> **"DNS request for all records.** This attack requests all records maintained on a remote server and it is used to gather information for a future attack." [NetRanger p. 4-64].

And then in SMTP:

> "**3103 Sendmail Reconnaissance**
> SubiD Values: 0
> Misc Field Info: 'vrfy' or 'expn'

Recommended Alarm Value: 2

This represents a reconnaissance attempt by an intruder. It may also represent an advanced user attempting to determine the mailing address of a friend or co-worker." *NetRanger User's Guide Version 1.3.1*, p. 4-73.

In short, the prior-art shows pervasive use of monitoring of data transfer commands in network traffic by intrusion detection systems, and thus it is an obvious extension of NSM to consider them. Therefore claim 2 is invalid.

4. The method of claim 1, wherein the measure monitors data transfers by monitoring network packet data transfer volume.

The NSM clearly discloses monitoring the network packet data transfer volume on network connections. Specifically, on p300 of [Heb90], we learn that "The counter in the cell indicates how many packets have been generated by this single connection", while on p301, we learn that "Examining the probability that each path will exist and the probability that the amount of traffic generated on each path is normal can be expensive, so the hierarchical search pattern is used to limit the depth of the search." And "the NSM's normal profile does not consist simply of a mean and variance; it consists of a range of values and the probability of observing a value at each range. Careful examination of network traffic showed that the data amounts were not always Gaussian distributed".

Clearly, the NSM was monitoring the volume of packets in connections including data transfers, and therefore falls in claim 4. Claim 4 is thereby invalid.

5. The method of claim 1, wherein the measure monitors network connections by monitoring network connection requests.

The NSM clearly falls under this claim. Network connections are a central part of its analysis. For example, see the quotes on the probability of network connections occurring in the analysis of claims 1 and 4. Therefore, claim 5 is invalid.

6. The method of claim 1, wherein the measure monitors network connections by monitoring network connection denials.

Again, both of the leading commercial intrusion detection systems at the time of the statutory bar were reasoning based on network connection denials, and publicly disclosing this fact. It would therefore be obvious to one skilled in the art that this feature could be added to any network intrusion detection system.

Specific examples include, from RealSecure:

"IP Half Scan

> "A standard TCP connection is established by sending a SYN packet to the destination host. If the destination is waiting for a connection on the specified port, it will respond with a SYN/ACK packet. The initial sender then replies to the SYN/ACK with an ACK packet, and the connection is established. If the destination host is not waiting for a connection on the specified port, it will respond with an RST packet instead of a SYN/ACK. Most system logs do not log completed connections until the final ACK packet is received from the source. Sending an RST packet instead of the final ACK results in the connection never actually being established; so no logging takes place. Because the source can identify whether the destination host sent a SYN/ACK or an RST, an attacker can determine exactly what ports are open for connections, without the destination ever being aware of probing." [RealSecure p. A-5] [RealSecure 1.1].

This text discloses ISS monitoring to see whether or not a connection succeeded by looking at RST packets, and capable of forming an alert based on the denials.

And from NetRanger

> "TCP Connection Logging…Level 2: Only TCP SYN packets are logged (default). This indicates that the source host has initiated an attempt to establish a TCP connection to the destination host using the TCP ports specified…. Level 3: All TCP, SYN, FIN, and RST packets are logged. The sequence numbers stored in the miscellaneous field provide enough information to determine the number of bytes transferred within a TCP connection." [NetRanger p. 4-71 – 4-72].

This text clearly indicates logging RST packets, which are indicative of network connection denials.

Since this was in wide practice, it would have been an obvious extension to NSM, and therefore claim 6 is invalid.

> *11. The method of claim 1, further comprising responding based on the determining whether the difference between the short-term statistical profile and the long-term statistical profile indicated suspicious network activity.*

To assess this, we clearly need to know what exactly responding is. SRI and Symantec are very broad in their proposed construction: "Taking an action in response". ISS is equally broad but spells out some possibilities: "taking an action in response to a suspected attack, including passive responses such as report dissemination to other monitors or administrators, and highly aggressive actions, such as severing a communication channel of the reconfiguration of logging facilities within network components."

There is support for the ISS version within the specification of the patent itself. In column 11, lines 32-43, we find:

> "Countermeasures range from very passive responses, such as report dissemination to other monitors 16a-16f or administrators, to highly aggressive actions, such as severing a communication channel or the reconfiguration of logging facilities within network components (e.g., routers, firewalls, network services, audit daemons). An active response may invoke handlers that validate the integrity of network services or other assets to ensure that privileged network services have not been subverted. Monitors 16a-16f may invoke probes in an attempt to gather as much counterintelligence about the source of suspicious traffic by using features such as traceroute or finger."

Since NSM was capable of "report dissemination to…administrators", it falls under this patent claim. [Heb90] specifically discloses of the user interface on p301:

> "It's purpose is to provide a user (e.g. the security officer) information about the Access Control Matric (ACM) in prictorial form that can be used to alert the security officer to attacks that change the tactics of the NSM."

Therefore, if we combine dependent claim 11 with independent claim 1 of the 338 patent, NSM falls entirely within it, and thus these claims are not novel over prior art.

> *12. The method of claim 11, wherein responding comprises transmitting an event record to a network monitor.*

It's obvious to combine NSM with [ref DIDS] which discloses a variant of the NSM reporting to the DIDS director, because they were combined in practice in the prior art. Therefore claim 12 is invalid.

> *13. The method of claim 12, wherein transmitting the event record to a network monitor comprises transmitting the event record to a hierarchically higher network monitor.*

It's obvious to combine NSM with [ref DIDS] which discloses a variant of the NSM reporting to the DIDS director, because they were combined in practice in the prior art. Therefore claim 13 is invalid.

> *14. The method of claim 13, wherein transmitting the event record to a network monitor comprises transmitting the event record to network monitor that receives event records from multiple network monitors.*

It's obvious to combine NSM with [ref DIDS] which discloses multiple instances of a variant of the NSM reporting to the DIDS director, because they were combined in practice in the prior art. Therefore claim 14 is invalid.

> *15. The method of claim 14, wherein the monitor that receives event records from multiple network monitors comprises a network monitor that correlates activity in the multiple network monitors based on the received event records.*

It's obvious to combine NSM with [ref DIDS] which discloses multiple instances of a variant of the NSM reporting to the DIDS director, which correlates information from them. See discussion under 203 vs DIDS claim 2 for more detail.

16. The method of claim 11, wherein responding comprises altering analysis of the network packets.

[Heb90] discloses on p 301

"It's purpose is to provide a user (e.g. the security officer) information about the Access Control Matric (ACM) in pictorial form that can be used to alert the security officer to attacks that change the tactics of the NSM."

"change the tactics" discloses altering analysis.

18. The method of claim 1, wherein the network packets comprise TCP/IP packets.

NSM exclusively analyzed TCP/IP packets. Therefore this claim is invalid.

19. The method of claim 1, wherein the network entity comprises a gateway, a router, or a proxy server.

The deployment of intrusion detection algorithms in at least routers was under broad public consideration in the intrusion detection community prior to November 1997, including and especially at DARPA principal investigator meetings where SRI staff were present. For example, the IDIP system [ref IDIP] had modules specifically designed to operate in routers and have the routers collaborate to trace the source of problems and block connections from systems that had been identified as intrusive. The JiNao system had modules intended to work inside routers [ref JiNao]. Chen and Levitt described a protocol for doing intrusion detection in routers to detect misbehaving ones. That paper was published in September 1997 [Che97]. SRI themselves disclosed the possibility prior to the statutory bar. For example, in the 1997 NISSC paper they write on page 355:

"Service monitors are dynamically deployed within a domain to provide localized real-time analysis of infrastructure (e.g., routers or gateways)..."

Thus it would have been obvious to one skilled in the art in 1997 that NSM could incorporate packet data captured on a router. With these obvious extensions, NSM is a system that would fall under claim 1 as refined by claim 19 of the 338 patent. Thus this is not novel over prior art.

24. A computer program product, disposed on a computer readable medium, the product including instructions for causing a processor to:
receive network packets handled by a network entity;
build at least one long-term and at least one short-term statistical profile from at least one measure of the network packets, the at least one measure monitoring data transfers, errors, or network connections.

compare at least one short-term and at least one long-term statistical profile; and
determing whether the difference between the short-term statistical profile and the long-term statistical profile indicates suspicious network activity.

This claim does not materially differ from claim 1, and the user is referred to the discussion there. Claim 24 is invalid for the same reasons.

25. A method of network surveillance comprising:
receiving packets at a network at a virtual private network entity; and
building at least one long-term and at least one short-term statistical profile based on the received packets, and.
comparing at least one long-term statistical profile with at least one short-term statistical profile to determine whether packets indicate suspicious network activity.

It is obvious that any monitoring technique that can be applied to a network can be applied to a VPN. That's what the "Virtual" in "Virtual Private Network" means. Therefore, this claim does not materially differ from claim 1, and the user is referred to the discussion there. Claim 25 is invalid for the same reasons.

Response to SRI contentions on NSM and the 338 patent.

In their response to ISS's second interrogatories on pp 12-14, SRI assert that NSM did not anticipate any claim of the 338 patent.

However, as we discussed at length in our consideration of the 338 patent above, it is obvious to combine the NSM reference with IDES since the text of the NSM reference specifically suggests doing so:

> "Therefore, for the present time, we have followed a similar path as that by IDES [9] and Wisdom and Sense [19], viz. we generate a mask of the normal traffic..."

and

> "We are currently looking at generating masks using the techniques used by IDES..."

Thus SRI needs to respond to this combination.

I further believe that a number of SRI's specific contentions in their response are inaccurate. For example, SRI asserts that (p13)

> "The NSM reference however provides no details as to how the matrix of normal behavior is constructed and nothing in the citations asserted by Defendants teaches or suggests that this matrix is a long-term statistical profile as recited in the claims."

To the contrary, NSM provides considerable detail, to the point where I feel confident I could build a system of substantially equivalent functionality from the description in the

paper. This is true of the NSM reference alone. It is also true when combined with IDES, as suggested by the NSM paper (and as a reference that one skilled in the art would certainly have been very aware of in 1997).

We need to distinguish between the "current traffic matrix", and the "normal traffic mask". The two data structures are related, but it is the latter that performs the role of the long-term profile.

Specific relevant text on the construction of the normal traffic mask, and its nature as a statistical profile as is follows.

An overview of the structure of matrices is given on p 298:

> "This section presents the conceptual view of the NSM. Currently, the NSM uses a four dimensional matrix of which the axes are: Source (a host which generates traffic), Destination (a host which traffic is destined), Service (mail, login, etc.), and Connection ID (a unique identified for a specific connection). Each cell in the matrix represents a unique connection on the network from a source host to a destination host by a specific service. This matrix is similar in concept to the well-known access matrix, the basis for protection in many systems. Each cell holds two values: the number of packets passed on the connection for a certain time interval, and the sum of the data carried by those packets..."

And details of the data structure are on p 300:

> "The linked-list matrix format consists of a list of nodes containing the address of hosts which have placed a packet on the network. Each of these "source" nodes has a list of nodes holding the addresses of hosts to which it, the source node, has sent a packet. Each of these "destination" nodes has a list of nodes holding information about each service used between the source and destination hosts. Finally, each connection node contains the number of packets used by the connection and which host, the source or destination, initiated the connection."

An overview of the nature of masks is on p 299:

> "A mask is a representation of the value for traffic measurements which are observed for some known traffic pattern. When the current traffic measurements are presented to the mask, only the measurements which have values matching the mask will pass through. A high percentage of the measurements passing through the mask would indicate that the traffic pattern for the mask is occurring."

With more detail later on the same page, which discloses the nature of masks as a statistical profile:

> "The actual representation of the mask can be performed by several methods. We currently generate a probabilistic distribution of values for each measurement which is seen frequently. The value for a current measurement is compared to the distribution, and if the probability of that value occurring is too low, it is passed through the mask."

Further evidence that the normal profile is a "statistical profile" is on p301.

> "Finally, the NSM's normal profile does not consist simply of a mean and variance; it consists of a range of values and the probability of observing a value at each range. Careful examination of network traffic showed that data amounts were not always Gaussian distributed; therefore, the mean and variance could not capture the true shape of the data."

The comparison of mask and matrix is discussed on p 300:

> "Detecting unusual patterns by a probabilistic analysis of the traffic requires knowledge of what the normal traffic flow is. The current traffic matrix is then compared to the normal/abnormal traffic mask to determine if something unusual is happening."

Also, the comparison at the cell level of normal traffic mask and current traffic matrix is described on p301:

> "The prototype examines the current network traffic when a new node is added, and at five minute intervals. When a new node is added to the network, the probabilistic analysis examines the cell against the normal/abnormal mask. At five minute intervals, the entire traffic matrix is compared to the normal/abnormal mask and the rules. Examining the probability that each path will exist and the probability that the amount of traffic generated on each path is normal can be expensive, so the hierarchical search pattern is used to limit the depth of the search."

The construction of the "normal/abnormal traffic mask" is covered in section 5.1 of [ref NSM]:

> "Finally, the matrix archiver writes the matrix representing the current traffic out to disk. Currently, a signal to save the matrix is sent to the archiver by the matrix analyzer every fifteen minutes… The archived files can be used to build or update a network profile."

Here we have a full page of quotations from the reference which to me very clearly disclose and enable a scheme in which a four-dimensional nested linked-list of recent connections, with their traffic volume (by both packets and bytes), is compared to a nested linked list recording the probability of any given connection occurring on a particular day, and the probability distribution for the number of packets and bytes (the byte count comparison is indicated to not have been implemented in the NSM, but is an easy extension once the code for anomaly detection on the packet count is done). Any person skilled in the art should be able to implement based on this description, and the normal traffic mask is clearly a long-term statistical profile, as it is storing a statistical probability distribution for values for use in comparison (much as IDES/NIDES profiles do).

Additionally, SRI asserts with little or no supporting argument that the NSM does not disclose the restrictions of the various dependent claims. See the claim-by-claim analysis above for details of my analysis of each dependent claim.